

# fifo.v

---

## AUTHORS

---

## JAY CONVERTINO

---

## DATES

---

2021/06/29

---

## INFORMATION

---

### Brief

---

Wrapper to tie together fifo\_ctrl, fifo\_mem, and fifo\_pipe.

### License MIT

---

Copyright 2021 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## fifo

---

```
module fifo #(
  parameter
    FIFO_DEPTH
    =
    256,
  parameter
    BYTE_WIDTH
    =
    1,
  parameter
    COUNT_WIDTH
```

```

    =
    8,
    parameter
    FWFT
    =
    0,
    parameter
    RD_SYNC_DEPTH
    =
    0,
    parameter
    WR_SYNC_DEPTH
    =
    0,
    parameter
    DC_SYNC_DEPTH
    =
    0,
    parameter
    COUNT_DELAY
    =
    1,
    parameter
    COUNT_ENA
    =
    1,
    parameter
    DATA_ZERO
    =
    0,
    parameter
    ACK_ENA
    =
    0,
    parameter
    RAM_TYPE
    =
    "block"
) ( input rd_clk, input rd_rstn, input rd_en, output rd_valid, output [(BYT

```

Wrapper to tie together fifo\_ctrl, fifo\_mem, and fifo\_pipe.

## Parameters

<b>FIFO_DEPTH</b> parameter	Depth of the fifo, must be a power of two number(divisable aka $256 = 2^8$ ). Any non-power of two will be rounded up to the next closest.
<b>BYTE_WIDTH</b> parameter	How many bytes wide the data in/out will be.
<b>COUNT_WIDTH</b> parameter	Data count output width in bits. Should be the same power of two as fifo depth(256 for fifo depth... this should be 8).
<b>FWFT</b> parameter	1 for first word fall through mode. 0 for normal.
<b>RD_SYNC_DEPTH</b> parameter	Add in pipelining to read path. Defaults to 0.
<b>WR_SYNC_DEPTH</b> parameter	Add in pipelining to write path. Defaults to 0.
<b>DC_SYNC_DEPTH</b> parameter	Add in pipelining to data count path. Defaults to 0.
<b>COUNT_DELAY</b> parameter	Delay count by one clock cycle of the data count clock. Set this to 0 to disable (only disable if read/write/data_count are on the same clock

	domain!).
<b>COUNT_ENA</b> parameter	Enable the count output.
<b>DATA_ZERO</b> parameter	Zero out data output when enabled.
<b>ACK_ENA</b> parameter	Enable an ack when data is requested.
<b>RAM_TYPE</b> parameter	Set the RAM type of the fifo.

## Ports

<b>rd_clk</b>	Clock for read data
<b>rd_rstn</b>	Negative edge reset for read.
<b>rd_en</b>	Active high enable of read interface.
<b>rd_valid</b>	Active high output that the data is valid.
<b>rd_data</b>	Output data
<b>rd_empty</b>	Active high output when read is empty.
<b>wr_clk</b>	Clock for write data
<b>wr_rstn</b>	Negative edge reset for write
<b>wr_en</b>	Active high enable of write interface.
<b>wr_ack</b>	Active high when enabled, that data write has been done.
<b>wr_data</b>	Input data
<b>wr_full</b>	Active high output that the FIFO is full.
<b>data_count_clk</b>	Clock for data count
<b>data_count_rstn</b>	Negative edge reset for data count.
<b>data_count</b>	Output that indicates the amount of data in the FIFO.

## INSTANTIATED MODULES

---

### pipe

---

```
fifo_pipe #(
    RD_SYNC_DEPTH(RD_SYNC_DEPTH),
    WR_SYNC_DEPTH(WR_SYNC_DEPTH),
    DC_SYNC_DEPTH(DC_SYNC_DEPTH),
    BYTE_WIDTH(BYTE_WIDTH),
    DATA_ZERO(DATA_ZERO),
    COUNT_WIDTH(COUNT_WIDTH)
) pipe ( .rd_clk(rd_clk), .rd_rstn(rd_rstn), .rd_en(rd_en), .rd_valid(s_rd_v
```

Pipe for data sync/clock issues.

## control

---

```
fifo_ctrl #(  
    FIFO_DEPTH(c_FIFO_DEPTH),  
    BYTE_WIDTH(BYTE_WIDTH),  
    ADDR_WIDTH(c_PWR_FIFO),  
    COUNT_WIDTH(COUNT_WIDTH),  
    COUNT_DELAY(COUNT_DELAY),  
    COUNT_ENA(COUNT_ENA),  
    ACK_ENA(ACK_ENA),  
    FWFT(FWFT)  
) control ( .rd_clk(rd_clk), .rd_rstn(rd_rstn), .rd_en(s_rd_en), .rd_addr(s_
```

Block RAM control, so it will act like a FIFO.

## inst\_dc\_block\_ram

---

```
dc_block_ram #(  
    RAM_DEPTH(c_FIFO_DEPTH),  
    BYTE_WIDTH(BYTE_WIDTH),  
    ADDR_WIDTH(c_PWR_FIFO),  
    RAM_TYPE(RAM_TYPE)  
) inst_dc_block_ram ( .rd_clk(rd_clk), .rd_rstn(rd_rstn), .rd_en(s_rd_mem_en)
```

Block RAM